# IDO: Intelligent Data Outsourcing with Improved RAID Reconstruction Performance in Large-Scale Data Centers

*Suzhen Wu*[12]*, Hong Jiang*[2]*, Bo Mao*[2]
[1]*Computer Science Department, Xiamen University*
[2]*Department of Computer Science & Engineering, University of Nebraska-Lincoln*
*suzhen@xmu.edu.cn, {jiang, bmao}@cse.unl.edu*

## Abstract

Dealing with disk failures has become an increasingly common task for system administrators in the face of high disk failure rates in large-scale data centers consisting of hundreds of thousands of disks. Thus, achieving fast recovery from disk failures in general and high on-line RAID-reconstruction performance in particular has become crucial. To address the problem, this paper proposes *IDO* (*Intelligent Data Outsourcing*), a proactive and zone-based optimization, to significantly improve on-line RAID-reconstruction performance. IDO moves popular data zones that are proactively identified in the normal state to a surrogate set at the onset of reconstruction. Thus, IDO enables most, if not all, user I/O requests to be serviced by the surrogate set instead of the degraded set during reconstruction.

Extensive trace-driven experiments on our lightweight prototype implementation of IDO demonstrate that, compared with the existing state-of-the-art reconstruction approaches WorkOut and VDF, IDO simultaneously speeds up the reconstruction time and the average user response time. Moreover, IDO can be extended to improving the performance of other background RAID support tasks, such as re-synchronization, RAID reshape and disk scrubbing.

## 1 Introduction

RAID [16] has been widely deployed in large-scale data centers owing to its high reliability and availability. For the purpose of data integrity and reliability, RAID can recover the lost data in case of disk failures, a process also known as *RAID reconstruction*. With the growing number and capacity of disks in data centers, the slow performance improvement of the disks and the increasing disk failure rate in such environments [18, 20], the RAID reconstruction is poised to become the norm rather than the exception in large-scale data centers [2,5,6]. Moreover, it was also pointed out that the probability of a second disk failure in a RAID system during reconstruction increases with the reconstruction time: approximately 0.5%, 1.0% and 1.4% for one hour, 3 hours and 6 hours of reconstruction time, respectively [6]. If another disk failure or latent sector errors [3] occur during RAID5 reconstruction, data will be lost, which is unacceptable for end users and makes the use of RAID6 more necessary and urgent. Therefore, the performance of on-line RAID reconstruction is of great importance to the reliability and availability of large-scale RAID-structured storage systems.

A number of optimizations have been proposed to improve the on-line RAID-reconstruction performance [8, 12, 21–24, 26, 28–31]. However, all of them are failure-induced or *reactive* optimizations and thus passive. In other words, they are triggered *after* a disk failure has been detected and focus on either improving the reconstruction workflow [23, 26, 30] or alleviating the user I/O intensity during RAID reconstruction [24, 28, 29] but *not both*. In fact, our workload analysis reveals that the reactive optimization is far from being adequate (see Section 2.3 for details).

On the other hand, from extensive evaluations and analysis, our previous studies and research by others have found that user I/O intensity during reconstruction has a significant impact on the on-line RAID-reconstruction performance because there are mutually adversary impacts between reconstruction I/O requests and user I/O requests [23, 24, 28]. This is why the time spent on the on-line RAID reconstruction is much longer than that on its off-line counterpart [7]. However, existing on-line RAID reconstruction approaches, such as WorkOut and VDF, only exploit the temporal locality of workloads to reduce the user I/O requests during reconstruction, which results in very poor reconstruction performance under workloads with poor temporal locality, as clearly evidenced in the results under the Microsoft Project trace that lacks temporal locality (see Section 4

for details). Therefore, we strongly believe that both the temporal locality and spatial locality of user I/O requests must be simultaneously exploited to further improve the on-line RAID-reconstruction performance.

Based on these observations, we propose a novel reconstruction scheme, called *IDO* (Intelligent Data Outsourcing), to significantly improve the on-line RAID-reconstruction performance in large-scale data centers by proactively exploiting data access patterns to judiciously outsource data. The main idea of IDO is to divide the entire RAID storage space into zones and identify the popularity of these zones in the normal operational state, in anticipation for data reconstruction and migration. Upon a disk failure, IDO reconstructs the lost data blocks belonging to the hot zones prior to those belonging to the cold zones and, at the same time, migrates these fetched hot data to a surrogate RAID set (*i.e.*, a set of spare disks or free space on another live RAID set [28]). After all data in the hot zones is migrated, most subsequent user I/O requests can be serviced directly by the surrogate RAID set instead of the much slower degraded RAID set. By simultaneously optimizing the reconstruction workflow and alleviating the user I/O intensity, the reconstruction speed of the degraded RAID set is accelerated and the user I/O requests are more effectively serviced, thus significantly reducing both the reconstruction time and the average user response time.

The technique of data migration has been well studied for performance improvement [1, 10, 11] and energy efficiency [17, 25] of storage systems, IDO adopts this technique in a unique way to significantly optimize the increasingly critical RAID reconstruction process in large-scale data centers. Even though IDO works for all RAID levels, we have implemented the IDO prototype by embedding it into the Linux software RAID5/6 module as a representative case study to assess IDO's performance and effectiveness. The extensive trace-driven evaluations show that IDO speeds up WorkOut [28] by a factor of up to 2.6 with an average of 2.0 in terms of the reconstruction time, and by a factor of up to 1.7 with an average of 1.3 in terms of the average user response time. IDO speeds up VDF [24] by a factor of up to 4.1 with an average of 3.0 in terms of the reconstruction time, and by a factor of up to 3.7 with an average of 2.3 in terms of the average user response time.

More specifically, IDO has the following salient features:

- IDO is a *proactive* optimization that dynamically captures the data popularity in a RAID system during the normal operational state.
- IDO exploits both *the temporal locality and spatial locality* of workloads on all disks to improve the on-line RAID-reconstruction performance.
- IDO optimizes both the reconstruction workflow and user I/O intensity to improve the RAID-reconstruction performance.
- IDO is simple and independent of the existing RAID tasks, thus it can be easily extended to improve the performance of other background tasks, such as re-synchronization, RAID reshape and disk scrubbing.

The rest of this paper is organized as follows. Background and motivation are presented in Section 2. We describe the design of IDO in Section 3. Methodology and results of a prototype evaluation of IDO are presented in Section 4. The main contributions of this paper and directions for the future research are summarized in Section 5.

## 2 Background and Motivation

In this section, we provide the necessary background about RAID reconstruction and key observations that motivate our work and facilitate our presentation of IDO in the later sections.

### 2.1 RAID reconstruction

Recent studies of field data on partial or complete disk failures in large-scale data centers indicate that disk failures happen at a higher rate than expected [3, 18, 20]. Schroeder & Gibson [20] found that annual disk replacement rates in the real world exceed 1%, with 2%-4% on average and up to 13% in some systems, much higher than 0.88%, the annual failure rates (AFR) specified by the manufacturer's datasheet. Bairavasundaram et al. [3] observed that the probability of latent sector errors, which can lead to disk replacement, is 3.45% in their study. The high disk failure rates, combined with the continuously increasing number and capacity of drives in large-scale data centers, are poised to render the reconstruction mode the common mode, instead of the exceptional mode, of operation in large-scale data centers [5, 6].

Figure 1 shows an overview of on-line reconstruction for a RAID5/6 disk array that continues to serve the user I/O requests in a degraded mode. The RAID-reconstruction thread issues reconstruction I/O requests to all the surviving disks and rebuilds the data blocks of the failed disk to the new, replacement disk. In the meantime, the degraded RAID5/6 set must service the user I/O requests that are evenly distributed to all the surviving disks. Thus, during on-line RAID reconstruction, reconstruction requests and user I/O requests will compete for

the bandwidth of the surviving disks and adversely affect each other. User I/O requests delay the reconstruction process while the reconstruction process increases the user response time. Previous studies [23, 24, 28] have demonstrated that reducing the amount of user I/O traffic directed to the degraded RAID set is an effective approach to simultaneously reducing the reconstruction time and alleviating the user performance degradation, thus improving both reliability and availability of storage systems.

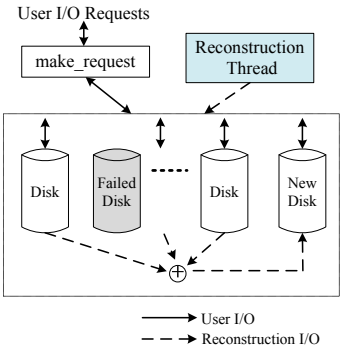



Figure 1: An overview of on-line reconstruction process for a RAID5/6 disk array.

## 2.2 Existing reconstruction approaches

Since RAID [16] was proposed, a rich body of research on the on-line RAID reconstruction optimization has been reported in the literature. Generally speaking, these approaches can be categorized into two types: optimizing the reconstruction workflow and optimizing the user I/O requests.

The first type of reconstruction optimization methods improve performance by adjusting the reconstruction workflow. Examples of this include SOR [9], DOR [8], PR [12], Live-block recovery [21], PRO [23], and JOR [27]. DOR [8] assigns one reconstruction thread for each disk, unlike SOR [9] that assigns one reconstruction thread for each stripe, allowing DOR to efficiently exploit the disk bandwidth to improve the RAID reconstruction performance. Live-block recovery [21] and JOR [27] exploit the data liveness semantics to reduce the RAID reconstruction time by ignoring the "dead" (no longer used) data blocks on the failed disk. PRO [23] exploits the user access locality by first reconstructing the hot data blocks on the failed disk. When the hot data blocks have been recovered, the reconstruction process for read requests to the failed disk can be significantly reduced, thus reducing both the reconstruction time and user I/O response time. Although the above reconstruction-workflow-optimized schemes can also improve the user I/O performance, the improvement is limited because the user I/O requests still must be serviced by the degraded RAID set. Therefore, the contention between user I/O requests and reconstruction I/O requests still persists.

The second type of reconstruction optimization methods improve performance by optimizing the user I/O requests. Examples of this include MICRO [30], WorkOut [28], Shaper [29], and VDF [24]. These optimization approaches directly improve the user I/O performance during reconstruction while simultaneously improving the reconstruction performance by allocating much more disk resources to the reconstruction I/O requests. MICRO [30] is proposed to collaboratively utilize the storage cache and the RAID controller cache to reduce the number of physical disk accesses caused by RAID reconstruction. VDF [24] improves the reconstruction performance by keeping the user requests belonging to the failed disk longer in the cache. However, if the requested data belonging to the failed disk have already been reconstructed to the replacement disk, the access delays of these user requests will not be further improved because they behave exactly the same as those of the data blocks belonging to the surviving disks [13]. Therefore, when VDF is incorporated into PRO [23], the improvement achieved by VDF will be significantly reduced. Both Shaper [29] and VDF [24] use the reconstruction-aware storage cache to selectively filter the user I/O requests, thus improving both the reconstruction performance and user I/O performance. Different from them, WorkOut [28] aims to alleviate the user I/O intensity on the entire degraded RAID set, not just the failed disk, during reconstruction by redirecting many user I/O requests to a surrogate RAID set.

While optimizing the user I/O requests can also reduce the on-line RAID reconstruction time, the performance improvement of the user-I/O-requests-optimized approaches above is limited. For example, both WorkOut and VDF only exploit the temporal locality of read requests, ignoring the beneficial spatial locality existing among read requests. Moreover, VDF gives higher priority to the user I/O requests addressed at the failed disk. However, the RAID reconstruction process involves all disks and the user I/O requests on the surviving disks also affect the reconstruction performance. And most importantly, the access locality tracking functions in these schemes are all initiated after a disk fails, which is passive and less effective.

## 2.3 Reactive vs. Proactive

The existing reconstruction optimizations initiate the reconstruction process only after a disk failure occurs, which we refer to as *failure-induced* or *reactive* optimizations, and thus are passive in nature. For example, PRO [23] and WorkOut [28] identify the popular data during reconstruction, which may result in insufficient

identification and exploitation of popular data. Compared with the normal operational state, the reconstruction period is too short to identify a sufficient amount of popular data by the reconstruction optimizations, as clearly evidenced by our experimental results in Section 2.5.

If the user I/O requests are monitored in the normal operational state, the popular data zones can be proactively identified before and in anticipation of a disk failure. Once a disk failure occurs, the optimization works immediately and efficiently by leveraging the data popularity information already identified. We call this process a *proactive* optimization, to contrast to its reactive counterpart. Figure 2 shows a comparison of user I/O performance between a reactive optimization and a proactive optimization, where the performance is degraded by the RAID reconstruction and returns to its normal level after completing the recovery. We can see that the proactive approach takes effect much faster than the reactive approach, shortening the reconstruction time. The reason is that the proactive approach with its popular data already accurately identified prior to the onset of the disk failure, can start reconstructing lost data immediately without the substantially extra amount of time required by the reactive approach to identify the popular data blocks.

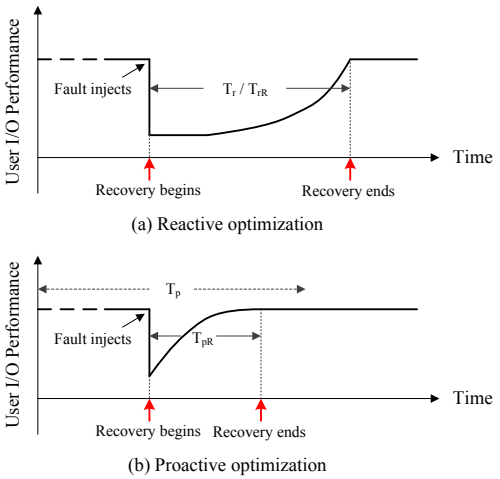



Figure 2: Comparisons of user I/O performance and reconstruction time between (a) reactive optimization and (b) proactive optimization. Note that $T_r$ and $T_p$ indicate the period of hot data identification, $T_{rR}$ and $T_{pR}$ denote the reconstruction time, and $T_r = T_{rR}$. In general, $T_r \ll T_p$ and $T_{rR} > T_{pR}$.

In large-scale data centers consisting of hundreds of thousands of disks, proactive optimization is very important because the disk-failure events are becoming the norm rather than the exception, for which RAID reconstruction is thus becoming a normal operation [5, 6].

## 2.4 Temporal locality vs. Spatial locality

In storage systems, access locality is reflected by the phenomenon of the same storage locations or closely nearby storage locations being frequently and repeatedly accessed. There are two dimensions of access locality. Temporal locality, on the time dimension, refers to the repeated accesses to specific data blocks within relatively small time durations. Spatial locality, on the space dimension, refers to the clustered accesses to data objects within small regions of storage locations within a short timeframe. These two access localities are the basic design motivations for storage-system optimizations.

Previous studies on RAID-reconstruction optimizations, such as VDF and WorkOut, use *request-based* optimization that only exploits the temporal locality of workloads, but not the spatial locality to reduce user I/O requests. PRO [23] and VDF [24] only focus on optimizing (*i.e.*, tracking or reducing) the user I/O requests to the failed disk, thus they ignore spatial locality and the impact of the user I/O requests on the surviving disks that also have notable performance impact on RAID reconstruction. Moreover, given the wide deployment of large-capacity DRAMs and flash-based SSDs as cache/buffer devices above HDDs to exploit temporal locality, the visible temporal locality at the HDD-based storage level is arguably very low. This is because of the filtering of the upper-level caches, while the visible spatial locality remains relatively high. The high cost of DRAMs and SSDs relative to that of HDDs makes good design sense for new system optimizations to put the large and sequential data blocks on HDDs for their high sequential performance, but cache the random and hot small data blocks in DRAMs and SSDs for their high random performance [4, 19]. As a result, these new system optimizations will likely render the existing temporal-locality-only RAID-reconstruction optimizations ineffective.

In order to capture both temporal locality and spatial locality to reduce the user I/O requests during reconstruction, we argue that *zone-based*, rather than *request-based*, data popularity identification and data migration schemes should be used. By migrating the "hot" and popular data zones to a surrogate RAID set immediately after a disk fails, it enables the subsequent user I/O requests to be serviced by the surrogate set during reconstruction. This improves the system performance by fully exploiting the spatial locality of the workload. In the meantime, the reconstruction process should rebuild the hot zones, rather than sequentially from the beginning to the end of the failed disk, to take the user I/O requests into consideration. By reconstructing the hot zones first, the data-migration overhead is reduced and most of the subsequent user I/O requests can be serviced by the surrogate set during reconstruction. In so doing,

the reconstruction workflow and the user I/O requests are simultaneously optimized to take the full advantages of both the temporal locality and spatial locality of workloads.

Figure 3 shows an example in which the request-based approach works in a reactive way by migrating the requested data on demand and thus fails to exploit the spatial locality. In contrast, the zone-based approach works in a proactive way by migrating the hot data zones, proactively identified during the normal operational state, to allow subsequent user read requests hit in the migrated data zones serviced by the surrogate set, thus further reducing the user I/O requests to the degraded RAID set during reconstruction.

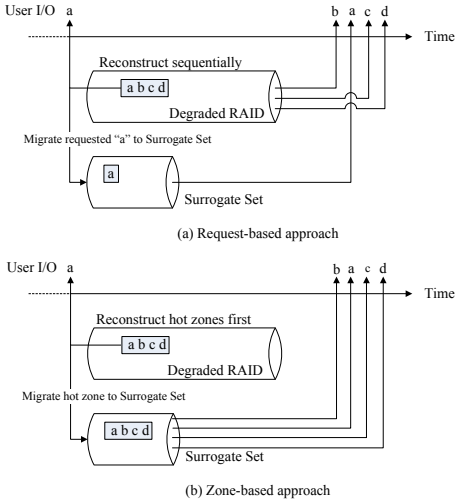



Figure 3: The I/O requests issued to the degraded RAID set with (a) a request-based approach and (b) a zone-based approach.

## 2.5 IDO motivation

Because user I/O intensity directly affects the RAID-reconstruction performance [28], we plot in Figure 4 the amount of user I/O traffic removed by a reactive request-based optimization (Reactive-request), a reactive zone-based optimization (Reactive-zone), a proactive request-based optimization (Proactive-request) and a proactive zone-based optimization (Proactive-zone), under the three representative traces, WebSearch2.spc, Financial2.spc and Microsoft Project. Each trace is divided into two disjoint parts, one runs in the normal operational state and the other runs in the reconstruction state. The reactive optimization, either request-based or zone-based, exploits the locality of user I/O requests in the reconstruction state and migrates the popular requests or zones to a surrogate set to allow the subsequent repeated read requests to be serviced by the surrogate set. The proactive scheme, either request-based or zone-based, exploits locality of user I/O requests by identifying the popular requests or hot zones in the normal operational state and migrating the hot requests or hot data zones to a surrogate set immediately after a disk fails, allowing any subsequent read requests that hit these migrated hot requests or zones to be serviced by the surrogate set during reconstruction.

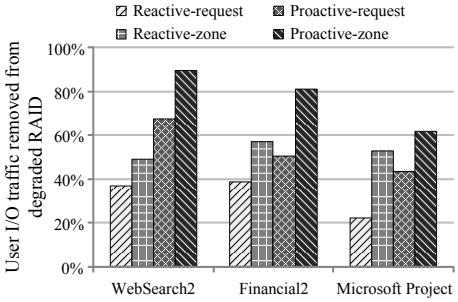



Figure 4: A comparison of the user I/O traffic removed from the degraded RAID set by a reactive request-based optimization (Reactive-request), a reactive zone-based optimization (Reactive-zone), a proactive request-based optimization (Proactive-request) and a proactive zone-based optimization (Proactive-zone), driven by three representative traces.

From Figure 4, we can see that the reactive-request scheme only exploits the data temporal locality in the reconstruction state, thus failing to remove a significant amount of user I/O traffic from the degraded RAID set. For traces with high spatial locality, such as the Microsoft Project trace, the reactive-zone scheme works better than the reactive-request scheme by removing an additional 30.5% of user I/O traffic from the degraded RAID set. For traces with high locality, be it temporal locality or spatial locality, the proactive approach removes much more user I/O traffic than the reactive approach. For example, the proactive-request scheme removes 41.4% and 21.2% more user I/O traffic than the reactive-request scheme for the WebSearch2.spc and Microsoft Project traces, respectively. By combining the proactive and zone approaches, the proactive-zone scheme removes the highest amount of the user I/O traffic from the degraded RAID set, with up to 89.8%, 81.2%, and 61.9% of the user I/O traffic being removed during reconstruction for the WebSearch2.spc, Financials.spc and Microsoft Project traces, respectively.

Clearly, the proactive optimization is much more efficient and effective than its reactive counterpart. Moreover, exploiting both the temporal locality and spatial locality (*i.e.*, zone-based) is better than exploiting only the temporal locality (*i.e.*, request-based), especially for the HDD-based RAIDs in the new HDD/SSD hybrid storage systems. If the hot data zones have been identified in the normal operational state (*i.e.*, without any disk failures)

and the data in these hot zones is migrated to a surrogate set at the beginning of the reconstruction period, the online RAID-reconstruction performance and user I/O performance can be simultaneously significantly improved.

Table 1 compares IDO with state-of-the-art reconstruction optimizations PRO, WorkOut and VDF based on several important RAID reconstruction characteristics. WorkOut [28] and VDF [24] only exploit the temporal locality of workloads to reduce the user I/O requests during reconstruction but ignore the spatial locality. PRO [23] and VDF [24] only focus on optimizing (*i.e.*, tracking or reducing) the user I/O requests to the failed disk but ignore the impact of the user I/O requests to the surviving disks that also have notable performance impact on RAID reconstruction. In contrast, IDO tracks all the user I/O requests addressed to the degraded RAID set in the normal operational state to obtain the data popularity information. Moreover, it exploits both the temporal locality and spatial locality of user I/O requests to both optimize the reconstruction workflow and alleviate the user I/O intensity to the degraded RAID set. Thus, both the RAID reconstruction performance and user I/O performance are simultaneously improved.

Table 1: Comparison of the reconstruction schemes.

| Characteristics | PRO [23] | WorkOut [28] | VDF [24] | IDO |
|---|---|---|---|---|
| Proactive | | | | ✓ |
| Temporal Locality | ✓ | ✓ | ✓ | ✓ |
| Spatial Locality | ✓ | | | ✓ |
| User I/O | | ✓ | ✓ | ✓ |
| Reconstruction I/O | ✓ | | | ✓ |

# 3 Intelligent Data Outsourcing

In this section, we first outline the main design objectives of IDO. Then we present its architecture overview and key data structures, followed by a description of the hot data identification, data reconstruction and migration processes. The data consistency issue in IDO is discussed at the end of this section.

## 3.1 Design objectives

The design of IDO aims to achieve the following three objectives.

- *Accelerating the RAID reconstruction performance* - By removing most of user I/O requests from the degraded RAID set, the RAID reconstruction process can be significantly accelerated.
- *Improving the user I/O performance* - By migrating the data belonging to the proactively identified hot zones to a surrogate RAID set, most subsequent user I/O requests can be serviced by the surrogate RAID set that is not affected by the RAID reconstruction process.
- *Providing high extendibility* - IDO is very simple and can be easily incorporated into the RAID functional module and extended into other background RAID tasks, such as re-synchronization, RAID reshape and disk scrubbing.

## 3.2 IDO architecture overview

IDO operates beneath the applications and above the RAID systems of a large data center consisting of hundreds or thousands of RAID sets, as shown in Figure 5. There are two types of RAID sets, *working RAID sets* and *surrogate RAID sets*. A working RAID set, upon a disk failure, becomes a *degraded RAID set* and is paired with a surrogate RAID set for the duration of reconstruction. A surrogate RAID set can be a dedicated RAID set that is shared by multiple RAID sets, or a RAID set that is capacity-shared with a lightly-loaded working RAID set. The dedicated surrogate RAID set improves the system performance but introduces extra device overhead, while the capacity-shared surrogate RAID set does not introduce extra device overhead but affects the performance of its own user applications. However, both are feasible and available for system administrators to choose from based on their characteristics and the system requirements. Moreover, the surrogate RAID set can be in the local storage node or a remote storage node connected by a network.

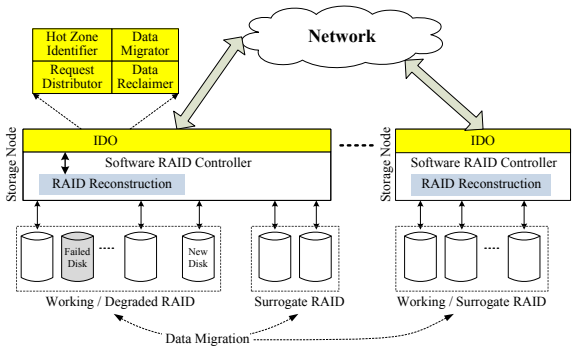



Figure 5: An architecture overview of IDO.

IDO consists of four key functional modules: Hot Zone Identifier, Request Distributor, Data Migrator and Data Reclaimer, as shown in Figure 5. *Hot Zone Identifier* is responsible for identifying the hot data zones in the RAID system based on the incoming user I/O requests. *Request Distributor* is responsible for directing the user I/O requests during reconstruction to the appropriate RAID set, *i.e.*, the degraded RAID set or the surrogate RAID set. *Data Migrator* is responsible for migrating all the data in the hot zones from the degraded RAID set to the surrogate RAID set, while *Data Reclaimer* is responsible for reclaiming all the redirected write data to

the newly recovered RAID set, *i.e.*, the previously degraded RAID set that has completed the reconstruction process. The detailed descriptions of these functional modules are presented in the following subsections.

IDO is an independent module added to an existing RAID system and interacts with its reconstruction module. In the normal operational state, only the Hot Zone Identifier module is active and tracks the popularity of each data zone. The other three modules of IDO remain inactive until the reconstruction module automatically activates them when the reconstruction thread initiates. They are deactivated when the reclaim process completes. The reclaim thread is triggered by the reconstruction module when the reconstruction process completes. IDO can also be incorporated into any RAID software to improve other background RAID tasks. In this paper, we mainly focus on the RAID reconstruction, but do include a short discusstion on how IDO works for some other background RAID tasks. This discusstion can be found in Section 4.4.

## 3.3 Key data structures

IDO relies on two key data structures to identify the hot data zones and record the redirected write data, namely, *Zone_Table* and *D_Map*, as shown in Figure 6. The Zone_Table contains the popularity information of all data zones, represented by three variables: *Num*, *Popularity* and *Flag*. *Num* indicates the sequence number of the data zone. Based on the Num value and the size of a data zone, IDO can calculate the start offset and end offset of the data zone to determine the target data zone for the incoming read request. *Popularity* indicates the popularity of the data zones. Its value is incremented when a read request hits the corresponding data zone. *Flag* indicates whether the corresponding data zone has been reconstructed and migrated. It is initialized to "00" and used by the Request Distributor module during the reconstruction period. The different values of Flag represent different states of the corresponding data zones, as shown in Figure 6.

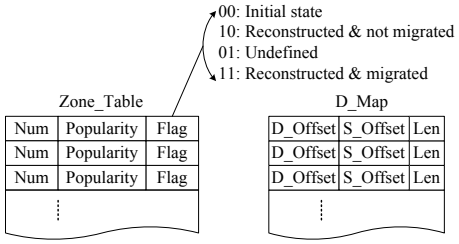



Figure 6: Main data structures of IDO.

The user read requests addressed to the data zones already migrated to the surrogate RAID set are redirected to it to reduce the user I/O traffic to the degraded RAID set. Besides that, IDO also redirects all user write requests to the surrogate RAID set to further alleviate the I/O intensity on the degraded RAID set. The D_Map records the information of all the redirected write data, including the following three variables. *D_Offset* and *S_Offset* indicate the offsets of the redirected write data on the degraded RAID set and the surrogate RAID set, respectively. *Len* indicates the length of the redirected write data. Similar to WorkOut [28], the redirected write data is sequentially stored on the surrogate RAID set to accelerate the write performance. Moreover, the redirected write data is only temporarily stored on the surrogate RAID set and thus should be reclaimed to the newly recovered RAID set after the reconstruction completes.

## 3.4 Hot data identification

IDO uses a dynamic hot data identification scheme, implemented in the Hot Zone Identifier module, to exploit both temporal locality and spatial locality of the user I/O requests. Figure 7 shows the hot data identification scheme in IDO. First, the entire RAID device space is split into multiple equal-size data zones of multiple-stripe size each. For example, in a 4-disk RAID5 set with a 64KB chunk size (*i.e.*, a stripe size of 3× 64KB = 192KB), the size of a data zone should be multiple times of 192KB. Therefore, a data zone is stripe aligned and can be fetched together to reconstruct the lost data blocks. Moreover, *spatial locality of requests* is also exploited since the tracking and migration unit of data is a data zone, thus IDO can capture the spatial locality of workloads by migrating data blocks prior to arrival of user I/O requests for those blocks. A detailed evaluation based on the selection of the different data zone sizes is presented in Section 4.3.

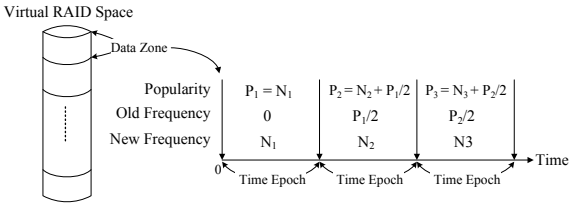



Figure 7: Hot data identification scheme in IDO. Note that the frequency of $N_*$ is defined as the number of user I/O requests issued to the data zone in a time epoch.

Second, in order to effectively and accurately exploit *temporal locality of requests*, the hot data zones in IDO are aged by decreasing their popularity values with time. More specifically, the popularity of a data zone in the current time epoch is calculated by first halving its popularity value from the previous time epoch before adding any values to it as more requests hit the zone in the current epoch. For example, as shown in Figure 7, $P_2$ is

equal to $N_2$ plus half of $P_1$ that is the popularity of the same data zone in its former time epoch. When a time epoch ends, the popularities of all data zones are halved in value.

Third, in order to reduce the impact of the hot data identification scheme on the system performance in the normal operational state, IDO updates the Zone_Table in the main memory without any disk I/O operations. When a read request arrives, IDO first checks its offset to locate the data zone that it belongs to. Then, IDO increments the corresponding *Popularity* in the Zone_Table in the main memory. Since the delay of the memory processing is much smaller than that of the disk I/O operation, the identification process in IDO has little performance impact on the overall system performance. Moreover, with the increasing processing power embedded in the storage controller, some storage systems already implement intelligent modules to identify the data popularity. Consequently, IDO can also simply utilize these functionalities to identify the hot data zones.

## 3.5 Data reconstruction and migration

Figure 8 shows the data reconstruction and migration processes in IDO. When a disk fails, the RAID reconstruction process is triggered with a hot spare disk in the degraded RAID set. IDO first reconstructs data in the hot data zones on the failed disk according to the Zone_Table. When the data blocks in the hot zones are read, IDO reconstructs and concurrently migrates all the data in these hot zones to the surrogate RAID set. Because the data is sequentially written on the surrogate RAID set, the overhead of the data migration, *i.e.*, writing the hot data to the surrogate RAID set, is minimal in IDO. When a hot data zone has been reconstructed and its data migrated to the surrogate RAID set, the corresponding *Flag* in the Zone_Table is set to "11". After all the hot data zones have been reconstructed, IDO begins to reconstruct the remaining data zones. In order to reduce the space overhead on the surrogate RAID set, IDO does not migrate the data in the cold data zones to the surrogate RAID set. Moreover, migrating the cold data does little to improve the overall system performance since few subsequent user I/O requests will be issued to these cold data zones. After a cold data zone has been reconstructed, the corresponding *Flag* in the Zone_Table is set to "10".

When all the data zones have been reconstructed, that is, the RAID reconstruction process is completed, the reclaim process for the redirected write data is initiated. In IDO, the redirected write data on the surrogate RAID set is protected by a redundancy scheme, such as RAID1 or RAID5/6. The priority of the reclaim process is set to be lower than the user I/O requests, which will not affect

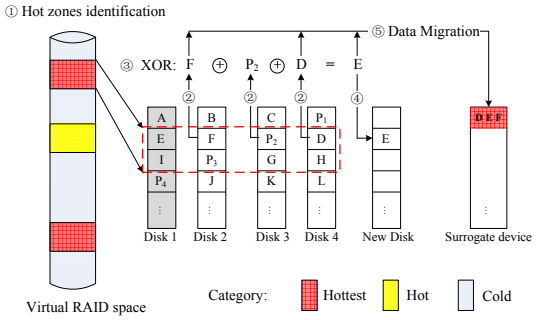



Figure 8: Data reconstruction and migration in IDO.

the reliability of the RAID system [28]. Therefore, the reclaim process for the redirected write data can also be scheduled in the system idle period. When a redirected write data block is reclaimed, its corresponding item in the D_Map is deleted. After all the items in the D_Map are deleted, the reclaim process completes.

During on-line RAID reconstruction, all incoming user I/O requests are carefully checked. Upon the arrival of a read request, IDO first determines its target data zone according to the Zone_Table and checks the second bit of the corresponding *Flag* to determine whether the data zone has been migrated or not ("1" indicates that the data zone has been migrated, while "0" indicates the opposite). If the data zone has not been migrated, the read request is issued to the degraded RAID set and the *Popularity* of the corresponding data zone is updated. Otherwise, the read request is issued to the surrogate RAID set. In order to obtain the accurate location on the surrogate RAID set, IDO checks the D_Map to determine whether the read request hits the previously redirected write data. If so, the read request is issued to the surrogate RAID set according to the *S_Offset* in the D_Map. Otherwise, the read request is issued to the surrogate RAID set according to the Zone_Table.

When processing a write request, the write data is sequentially written on the surrogate RAID set and IDO checks whether the write request hits the D_Map. If the write request hits the D_Map, the corresponding item in the D_Map is updated. Otherwise, a new item for the write request is added to the D_Map.

## 3.6 Data consistency

Data consistency in IDO includes two aspects: (1) The key data structures must be safely stored, (2) The redirected write data must be reliably stored on the surrogate set until the data reclaim process completes.

First, to prevent the loss of the key data structures in the event of a power supply failure or a system crash, IDO stores them in a non-volatile RAM (NVRAM). Since the size of Zone_Table and D_Map is generally very small, it will not incur significant extra hardware

cost. Moreover, in order to improve the write performance by using the write-back technique, the NVRAM is commonly deployed in the storage controllers. Thus, it is easy and reasonable to use the NVRAM to store the key data structures.

Second, the redirected write data must be safely stored on the surrogate set. To prevent data loss caused by a disk failure on the surrogate set, the surrogate set must be protected by a redundancy scheme, such as mirroring-based (RAID1) or parity-based (RAID5/6) disk arrays, a basic requirement for the surrogate RAID set. Our previous study [28] provides a detailed analysis on how to choose a surrogate set based on the requirements and characteristics of the applications. Moreover, since the up-to-date data for a read request can be stored on either the degraded RAID set or the surrogate set, each read request is first checked in the D_Map to determine whether it should be serviced by the degraded RAID set, the surrogate set or both (when the data is partially modified) to keep the fetched data always up-to-date, until all the redirected write data has been reclaimed.

# 4 Performance Evaluation

In this section, we present the performance evaluation of the IDO prototype through extensive trace-driven experiments.

## 4.1 Experimental setup and methodology

We have implemented an IDO prototype by embedding it into the Linux software RAID (MD) as a built-in module. IDO tracks the user I/O requests in the *make_request* function to identify the data popularity in the normal operational state. When a disk fails and the reconstruction thread is initiated by the *md_do_sync* function, the hot data zones are first reconstructed and migrated to the surrogate set. During reconstruction, the incoming user read requests are checked in the *make_request* function to determine by which device the requests are to be serviced, so as to avoid the degraded set whenever possible. All user write requests are issued to the surrogate set and marked as dirty for reclaim after the RAID reconstruction process completes.

The performance evaluation of IDO was conducted on a server-class hardware platform with an Intel Xeon X3440 processor and 8GB DDR memory. The HDDs are WDC WD1600AAJS SATA disks that were used to configure both the active RAID set and the surrogate RAID set. While the active set assumed a RAID5/6 organization, the surrogate set was configured as a RAID1 organization with 2 HDDs. Further, the surrogate set can be located either in the same storage node as the degraded RAID set or in a remote storage node in a data center. The rotational speed of these disks is 7200 RPM, with a sustained transfer rate of 60MB/s that is specified in the manufacture's datasheet. We used 10GB of the capacity of each disk for the experiments. A separate disk was used to house the operating system (Linux kernel version 2.6.35) and other software (MD and mdadm). In our prototype implementation, the main memory was used to substitute a battery-backed RAM for simplicity.

The traces used in our experiments were obtained from the UMass Trace Repository [15] and Microsoft [14]. The two financial traces (short for Fin1 and Fin2) were collected from the OLTP applications running at a large financial institution and the WebSearch2 trace (short for Web2) was collected from a machine running a web search engine. The Microsoft Project trace was collected in a volume storing the project directories (short for Proj). The four traces represent different access patterns in terms of read/write ratio, IOPS and average request size, with the main workload parameters summarized in Table 2.

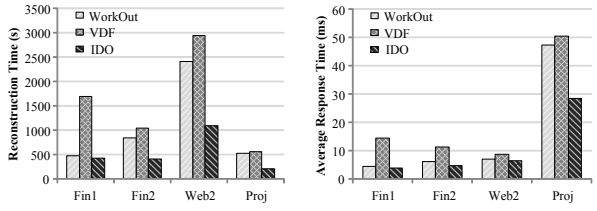
Table 2: The key evaluation workload parameters.

| Trace | Trace Characteristic | | |
|---|---|---|---|
| | Read Ratio | IOPS | Aver. Req. Size(KB) |
| Fin1 | 32.8% | 69 | 6.2 |
| Fin2 | 82.4% | 125 | 2.2 |
| Web2 | 100% | 113 | 15.1 |
| Proj | 97.6% | 29 | 57.8 |

To better examine the IDO performance under existing RAID reconstruction approaches, we incorporated IDO into MD's default reconstruction algorithm PR. We compared IDO with two state-of-the-art RAID reconstruction optimizations, WorkOut [28] and VDF [24], in terms of reconstruction performance and user I/O performance. WorkOut tracks the user access popularity and issues all write requests and popular read requests to the surrogate set during reconstruction. VDF exploits the fact that the user I/O requests addressed to the failed disk are expensive by keeping the requested data previously stored on the failed disk longer in the storage cache and choosing data blocks belonging to the surviving disks to evict first. Because VDF is a cache replacement algorithm, we applied it to the management of the surrogate set to make the comparison fair.

## 4.2 Performance results

We first conducted experiments on a 4-disk RAID5 set with a stripe unit size of 64KB while running WorkOut, VDF and IDO, respectively. Figure 9 shows the reconstruction time and average user response time under the minimum reconstruction bandwidth of 1MB/s, driven by the four traces. We configured a local 2-disk dedicated RAID1 set as the surrogate set to boost the reconstruction performance of the 4-disk degraded RAID5 set. For IDO, the data zone size was set to 12MB.




(a) Reconstruction Time (b) Average Response Time

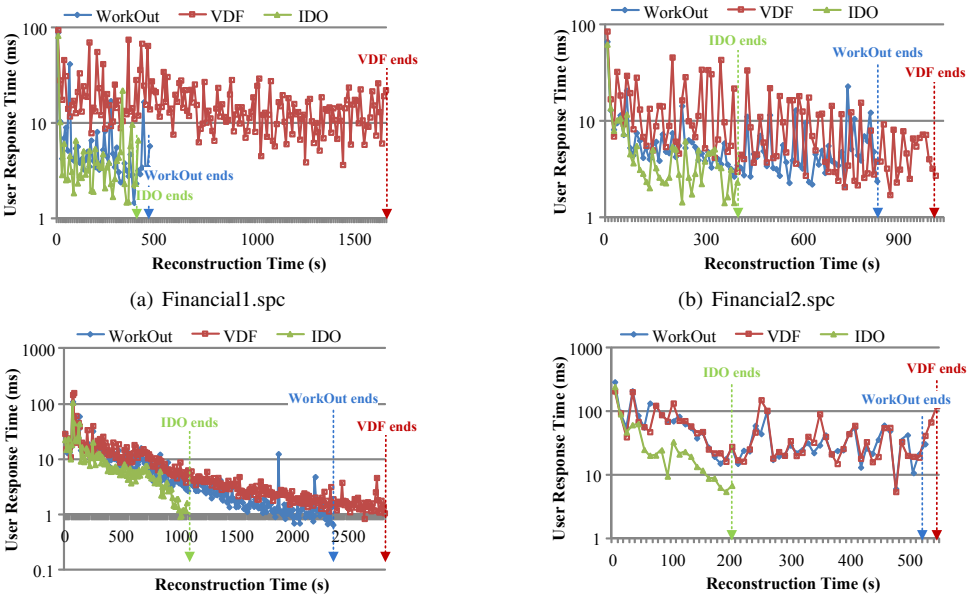
Figure 9: The performance results of WorkOut, VDF and IDO in a 4-disk RAID5 set with a stripe unit size of 64KB, 1MB/s minimum reconstruction bandwidth, and a local 2-disk dedicated RAID1 set as the surrogate RAID set, driven by the four traces.

From Figure 9(a), we can see that IDO speeds up WorkOut by a factor of 1.1, 2.1, 2.2 and 2.6, and speeds up VDF by a factor of 4.1, 2.5, 2.7 and 2.7 in terms of the reconstruction time for the Fin1, Fin2, Web2 and Proj traces, respectively. IDO's advantage stems from its ability to remove much more user I/O requests from the degraded RAID set than WorkOut and VDF, as indicated in Figure 11, which enables it to accelerate the RAID reconstruction process. However, since the Fin1 trace has much more write requests than read requests (as indicated in Table 2), IDO and WorkOut have similar abilities to remove the write requests from the degraded RAID set, reducing IDO's performance advantage over WorkOut. For the read-intensive traces, Fin2, Web2 and Proj, IDO removes much more read requests from the degraded RAID set than WorkOut. This is because IDO proactively identifies both the temporal locality and spatial locality in the normal operational state and migrates the hot data zones at the onset of reconstruction, while WorkOut only reactively identifies the temporal locality and migrates the user I/O requests after a disk fails. In this case, most subsequent read requests addressed to these hot data zones in IDO can be serviced directly by the surrogate RAID set instead of the much slower degraded RAID set. As a result, IDO's advantage margin over WorkOut is much wider under these three read-intensive traces than under the write-intensive Fin1 trace. For example, IDO reduces the reconstruction time much more significantly than WorkOut under the Proj trace because the Proj trace has poor temporal locality that leaves WorkOut much less room to improve than the spatial-locality-exploiting IDO.

On the other hand, from Figure 9(a), we can see that the performance of both WorkOut and IDO are better than that of VDF. The reason is that both WorkOut and IDO reduce not only the user I/O requests to the failed disk, but also the popular read requests and all write requests to the surviving disks. VDF keeps the data blocks belonging to the failed disk longer in the storage cache because servicing these data blocks is much more expensive than servicing the data blocks belonging to the surviving disks. However, if the data blocks belonging to the failed disk are already reconstructed, they will behave exactly the same way as the data blocks belonging to the surviving disks because the read redirection technique will fetch these data blocks directly from the replacement disk rather than reconstructing them from the surviving disks again. In the VDF evaluation reported in [24], the experimental kernel is Linux 2.6.32 without the read redirection function that has been implemented in the Linux MD software since Linux 2.6.35 [13]. The results also confirm the conclusion made in our previous WorkOut study that the user I/O intensity has a significant impact on the on-line RAID reconstruction performance.

Figure 9(b) shows that IDO speeds up WorkOut by a factor of 1.1, 1.3, 1.1 and 1.7, and speeds up VDF by a factor of 3.7, 2.4, 1.4 and 1.8 in terms of the average user response time (i.e., user I/O performance) during reconstruction for the Fin1, Fin2, Web2 and Proj traces, respectively. The reason why IDO only improves WorkOut slightly is that the minimum reconstruction bandwidth is set to be the default 1MB/s, which gives the user I/O requests a higher priority than the reconstruction requests. However, the performances of both IDO and WorkOut are better than that of VDF because IDO and WorkOut remove much more user I/O requests from the degraded RAID set than VDF, as revealed in Figure 11. Removing the user I/O requests from the degraded RAID set directly improves the user response time for the following two reasons. First, the response time of the redirected requests is no longer affected by the reconstruction process that competes for the available disk bandwidth with the user I/O requests on the degraded RAID set. Moreover, the redirected write data is sequentially laid out on the surrogate RAID set, thus further reducing the user response time. Second, many user I/O requests are removed from the degraded RAID set and the I/O queue on the degraded RAID set is accordingly shortened, thus reducing the response time of the remaining user I/O requests still serviced by the degraded RAID set.

Figure 10 compares in more details the reconstruction and use I/O performances of IDO, WorkOut and VDF during the reconstruction process, highlighting the significant advantage of IDO over WorkOut and VDF. Two aspects of the significant improvement in the user response time are demonstrated in these figures. First, the onset of the performance improvement of IDO is much earlier than that of WorkOut and VDF during reconstruction. The reason is that IDO has already captured the data popularity before a disk fails, thus enabling it to optimize the reconstruction process earlier and consequently outperform both WorkOut and VDF in terms of user response time during reconstruction. Second, IDO




(a) Financial1.spc

(b) Financial2.spc

(c) WebSearch2.spc

(d) MSR Project

Figure 10: A comparison of user response times of WorkOut, VDF and IDO during reconstruction in a 4-disk RAID5 set with a stripe unit size of 64KB, 1MB/s minimum reconstruction bandwidth, and a local 2-disk dedicated RAID1 set as the surrogate RAID set, driven by the four traces.

completes the reconstruction process much more quickly than WorkOut and VDF, which translates into improved reliability. The RAID reconstruction period is also called a "window of vulnerability" during which a subsequent disk failure (or a series of subsequent disk failures) will result in data loss [23]. Thus, shorter reconstruction time indicates higher reliability. Furthermore, user I/O requests in the IDO reconstruction experience a much shorter period of time in which they see increased response time than in the WorkOut and VDF reconstruction.

To gain a better understanding of the reasons behind the significant improvement achieved by IDO, we plotted the percentage of redirected requests for the three schemes under the minimum reconstruction bandwidth of 1MB/s. From Figure 11, we can see that IDO moves 88.1%, 78.7%, 72.4% and 42.0% of user I/O requests from the degraded RAID set to the surrogate RAID set for the four traces respectively, significantly more than either WorkOut or VDF does. This is because both WorkOut and VDF only exploit the temporal locality of user I/O requests and VDF only gives higher priority to the user I/O requests belonging to the failed disk. In contrast, IDO exploits both the temporal locality and spatial locality on all disks, reconstructs the hot data zones first and migrates them to the surrogate RAID set. And, most importantly, IDO uses a proactive optimization that is superior to the reactive optimizations, such as WorkOut and VDF. On the other hand, removing user I/O requests from the degraded RAID set directly reduces both the reconstruction time and user response time [28], something that IDO does much better than either WorkOut or VDF.

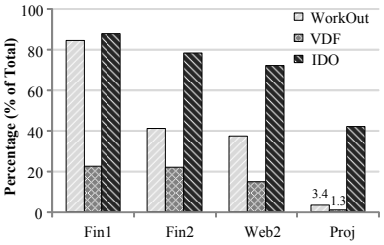
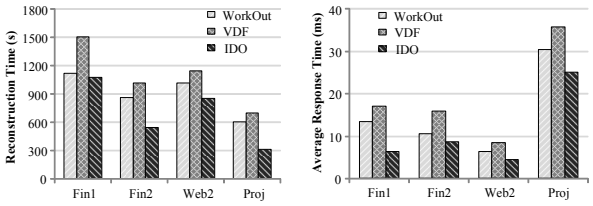



Figure 11: Percentage of redirected user I/O requests for WorkOut, VDF and IDO under the minimum reconstruction bandwidth of 1MB/s.

We also conducted experiments on a 6-disk RAID6 set (4 data + 2 parity) with a stripe unit size of 64KB under the minimum reconstruction bandwidth of 1MB/s. In the RAID6 experiments, we configured a 2-disk dedicated RAID1 set in the same storage node as the local surrogate set. In the experiments, we measured the reconstruction times and the average user response times when one disk fails.

From Figure 12, we can see that IDO improves both the reconstruction times and the average user response times over the WorkOut and VDF schemes. In particular, IDO speeds up WorkOut by a factor of up to 1.9 with an average of 1.4 in terms of the reconstruction time, and by a factor of up to 2.1 with an average of 1.5 in terms of the average user response time. IDO speeds up VDF

by a factor of up to 2.2 with an average of 1.7 in terms of the reconstruction time, and by a factor of up to 2.7 with an average of 2.0 in terms of the average user response time.




(a) Reconstruction Time (b) Average Response Time

Figure 12: Reconstruction times and average user response times of RAID6 reconstruction.

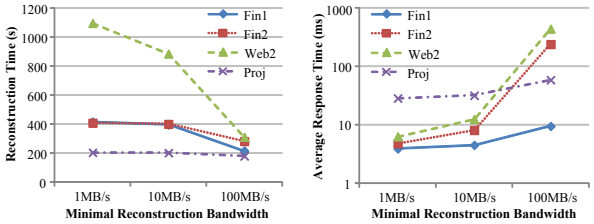
The reason behind the improvement on RAID6 is similar to that for RAID5. Upon a disk failure, all the disks in RAID6, as in RAID5, will be involved in servicing the reconstruction I/O requests. In the meantime, all the disks will service the user I/O requests. Thus, removing the user I/O requests can directly speed up the reconstruction process by allowing much more disk bandwidth to service the reconstruction I/O requests. Moreover, the average user response times also decrease because most of the user I/O requests are serviced on the surrogate RAID set without interfering with the reconstruction I/O requests. IDO works in a proactive way and exploits both the temporal locality and spatial locality of the user I/O requests, thus removing much more user I/O requests from the degraded RAID set to the surrogate set (as similarly indicated in the Figure 11) and performing better than Workout and VDF.

## 4.3 Sensitivity study

The IDO performance is likely influenced by several important factors, including the available reconstruction bandwidth, the data zone size, the stripe unit size, the number of disks, and the location of the surrogate set.

**Reconstruction bandwidth.** To evaluate how the minimum reconstruction bandwidth affects the reconstruction performance, we conducted experiments to measure reconstruction time and average user response time as a function of different minimum reconstruction bandwidths, 1MB/s, 10MB/s and 100MB/s, respectively. Figure 13 plotted the experimental results on a 4-disk RAID5 set with a stripe unit size of 64KB and a data zone size of 12MB.

From Figure 13(a), we can see that the reconstruction time generally decreases with the increasing minimum reconstruction bandwidth. However, for the Fin1, Fin2 and Proj traces, the reconstruction times remain almost unchanged when the minimum reconstruction bandwidth changes from 1MB/s to 10MB/s. The reason is that when the minimum reconstruction bandwidth is 1MB/s, the actual reconstruction speed is around 10MB/s due to the low user I/O intensity on the degraded RAID set. In contrast, From Figure 13(b), we can see that the user response time increases rapidly with the increasing minimum reconstruction bandwidth. When the minimum reconstruction bandwidth increases, much more reconstruction I/O requests are issued, thus lengthening the disk I/O queue and increasing the user I/O response time.



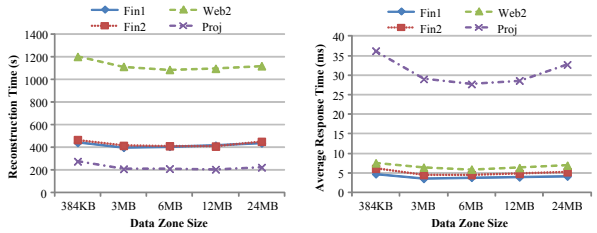

(a) Reconstruction Time (b) Average Response Time

Figure 13: Reconstruction times and average user response times of IDO as a function of different minimum reconstruction bandwidths (1MB/s, 10MB/s and 100MB/s) under the four traces.

**Data zone size.** In IDO, the data zone size is a key factor in identifying the data popularity. In order to evaluate the effect of the data zone size on the reconstruction performance and user I/O performance during reconstruction, we conducted experiments on a 4-disk RAID5 set with a stripe unit size of 64KB and different data zone sizes of 384KB, 3MB, 6MB, 12MB and 24MB, under the minimal reconstruction bandwidth of 1MB/s.

The results, shown in Figure 14, indicate that, with a very small data zone in one extreme, both the reconstruction time and the user response time are increased. The reason is that the reconstruction sequentiality is destroyed with a very small data zone. Moreover, the spatial locality is somewhat weakened with a very small zone size, although highly concentrated temporal locality is still captured. In the other extreme, with a very large data zone, the reconstruction performance and the user response time are again both increased. The reason is that with a very large data zone, IDO will likely migrate much more rarely-accessed cold data blocks to the surrogate set, thus wasting the disk bandwidth resources. Our experiments, driven by the four traces, seem to suggest that the data zone sizes between 3MB to 12MB are consistently the best.

**Stripe unit size.** To examine the impact of the stripe unit size on the RAID reconstruction, we conducted experiments on a 4-disk RAID5 set with stripe unit sizes of 4KB, 16KB and 64KB, respectively. The experimental results show that the reconstruction times and user response times are almost unchanged, suggesting that IDO

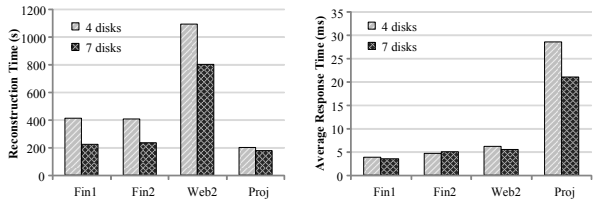



(a) Reconstruction Time (b) Average Response Time

Figure 14: Reconstruction times and average user response times of IDO as a function of different data zone size (384KB, 3MB, 6MB, 12MB and 24MB) under the four traces.

is not sensitive to the stripe unit size. The reason is that most user I/O requests are performed on the surrogate RAID set, thus the stripe unit size of the degraded RAID set has no impact on these redirected user I/O requests and very little impact overall. Accordantly, the RAID reconstruction speed is not affected. Due to space limits, these results are not shown here quantitatively.

**Number of disks.** To examine the sensitivity of IDO to the number of disks of the degraded RAID set, we conducted experiments on RAID5 sets consisting of different numbers of disks (4 and 7) with a stripe unit size of 64KB under the minimum reconstruction bandwidth of 1MB/s. From Figure 15, we can see that the reconstruction time decreases with the increasing number of disks. The reason is that the I/O intensity on individual disks will decrease when the RAID set has more disks, allowing for a shorter reconstruction time. However, the user I/O requests are not significantly affected since they are mostly serviced by the surrogate RAID set. The remaining user I/O requests still performed on the degraded RAID set only slightly affect the total user response time.

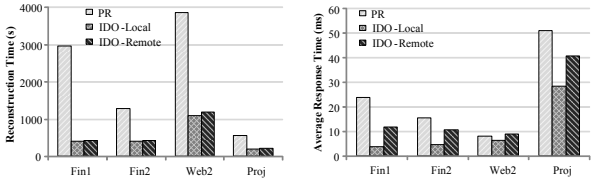



(a) Reconstruction Time (b) Average Response Time

Figure 15: Reconstruction times and average user response times of IDO as a function of different numbers of disks (4 and 7) under the four traces.

**Location of the surrogate device.** In a large-scale data center, the location of the surrogate set can also affect the RAID reconstruction performance. To examine the impact of the location of the surrogate set on the reconstruction performance, we conducted experiments to migrate the hot data to a different storage node connected with a gigabit Ethernet interface in a local area network.

Figure 16(a) shows that the reconstruction time is almost unchanged, which indicates that the reconstruction time is not sensitive to the location of the surrogate set. The reason is that no matter where the surrogate set is, the total numbers of redirected user I/O requests are the same, thus the reconstruction speed of the degraded RAID set is similar. On the other hand, the average user response time increases significantly with a remote surrogate set, as shown in Figure 16(b). The reason is that the response time of the redirected user I/O requests must now include the extra network delay with a remote surrogate set. However, compared with PR (the default reconstruction algorithm of Linux MD), IDO still significantly reduces the user response time and the reconstruction time with a remote surrogate set. The results suggest that in a large-scale data center, both the local and remote surrogate devices are helpful in improving the reconstruction performance, which further validates the effectiveness and adaptivity of IDO in large-scale data centers.

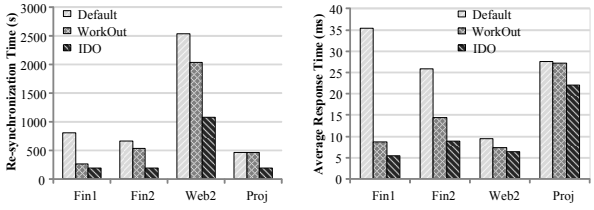



(a) Reconstruction Time (b) Average Response Time

Figure 16: Reconstruction times and average user response times with respect to different surrogate set locations under the four traces.

## 4.4 Extensibility

To demonstrate how IDO may be extended to optimize other background RAID tasks, we incorporated IDO into the RAID re-synchronization module. We conducted experiments on a 4-disk RAID5 set with a stripe unit size of 64KB under the minimum re-synchronization bandwidth of 1MB/s, driven by the four traces. We configured a dedicated 2-disk local RAID1 set as the surrogate set. The experimental results of the re-synchronization times and average user response times during re-synchronization are shown in Figure 17.




(a) Re-synchronization Time (b) Average Response Time

Figure 17: The re-synchronization results of the default re-synchronization function in the Linux MD software without any optimization, WorkOut and IDO.

Although the RAID re-synchronization process operates somewhat differently than the RAID reconstruction process, the re-synchronization requests compete for the disk resources with the user I/O requests during the on-line re-synchronization period in a way similar to the latter. By redirecting a significant amount of user I/O requests away from the RAID set undergoing re-synchronization, IDO can reduce both the re-synchronization time and user response time. The results are very similar to those in the above RAID reconstruction experiments, as are the reasons behind them.

## 4.5 Overhead analysis

Besides the device overhead for the surrogate set in the case of dedicated surrogate sets [28], we analyzed the following two overhead metrics in this paper: the performance overhead in the normal operational state and the memory overhead.

**Performance overhead.** Since IDO is a proactive optimization designed to improve the RAID reconstruction performance, it requires a hot data identification module that may affect the system performance in the normal operational state. In order to quantify how much the impact is, we conducted experiments to evaluate the user response times in the normal operational state with and without the hot data identification module.

From the experimental results, we find that the user response times in the two cases remain roughly unchanged under any of the four traces. In the worst case, the performance with the module activated degrades by less than 3% under the WebSearcch2.spc trace. The reason is that the hot data identification module only adds one extra operation (*i.e.*, incrementing the in-memory popularity value of the corresponding data zone by one) for each request. Thus the performance overhead is negligible in face of the high latency of the disk accesses.

**Memory overhead.** To prevent data loss, IDO uses non-volatile memory to store the Zone_Table and D_Map, thus incurring extra memory overhead. However, IDO uses less non-volatile memory capacity than WorkOut. The reason is that WorkOut migrates the user requested data to the surrogate set while IDO migrates the hot data zones. In WorkOut, each migrated write request and read request requires a corresponding entry in the mapping table. However, in IDO, only the migrated write requests and hot data zones require corresponding entries in the mapping information. Since the number of migrated hot data zones in IDO is generally much smaller than the number of hot read requests in WorkOut, the memory overhead of IDO is lower than that of WorkOut.

In the above experiments on the RAID5 set with individual disk capacity of 10GB, the maximum memory overheads are 0.11MB, 0.24MB, 0.11MB and 0.06MB for the Fin1, Fin2, Web2 and Proj traces, respectively. With the rapid increase in the size of memory and decrease in the cost of non-volatile memories, this memory overhead of IDO is arguably reasonable and acceptable to the end users.

# 5 Conclusion

In many data-intensive computing environments, especially data centers, large numbers of disks are organized into various RAID architectures. Because of the increased error rates for individual disk drives, the dramatically increasing size of drives, and the slow growth in transfer rates, the performance of RAID during its reconstruction phase (after a disk failure) has become increasingly important for system availability. We have shown that IDO can substantially improve this performance at low cost by using the free space available in these environments. IDO proactively exploits both the temporal locality and spatial locality of user I/O requests to identify the hot data zones in the normal operational state. When a disk fails, IDO first reconstructs the lost data blocks on the failed disk belonging to the hot data zones and concurrently migrates them to a surrogate RAID set. This enables most subsequent user I/O requests to be serviced by the surrogate RAID set, thus improving both the reconstruction performance and user I/O performance.

IDO is an ongoing research project and we are currently exploring several directions for future research. One possible direction is to design and conduct more experiments to evaluate the IDO prototype for other background tasks (such as RAID reshape and disk scrubbing) and other RAID levels (such as RAID10). Another is to build a power measurement module to evaluate the energy efficiency of IDO. By proactively migrating the hot data to the active RAID sets, the local RAID set can stay longer in the idle mode to save energy by redirecting read requests and write requests to the active RAID sets.

# Acknowledgments

We thank our shepherd Andrew Hume, our mentor Nicole Forsgren Velasquez, and the anonymous reviewers for their helpful comments. This work is supported by the National Science Foundation of China under Grant No. 61100033, the US National Science Foundation under Grant No. NSF-CNS-1116606, NSF-CNS-1016609, NSF-IIS-0916859, and NSF-CCF-0937993.

# References

[1] R. Arnan, E. Bachmat, T. Lam, and R. Michel. Dynamic Data Reallocation in Disk Arrays. *ACM-Transactions on Storage*, 3(1):2, 2007.

[2] Storage at Exascale: Some Thoughts from Panasas CTO Garth Gibson. Interview. http://www.hpcwire.com/hpcwire/2011-05-25/storage_at_exascale_some_thoughts_from_panasas_cto_garth_gibson.html. May. 2011.

[3] L. N. Bairavasundaram, G. R. Goodson, S. Pasupathy, and J. Schindler. An Analysis of Latent Sector Errors in Disk Drives. In *SIGMETRICS'07*, Jun. 2007.

[4] F. Chen, David A. Koufaty, and X. Zhang. Hystor: Making the Best Use of Solid State Drives in High Performance Storage Systems. In *ICS'11*, Jun. 2011.

[5] Veera Deenadhayalan. GPFS Native RAID for 100,000-Disk Petascale Systems. In *LISA'11*, Dec. 2011.

[6] G. Gibson. Reflections on Failure in Post-Terascale Parallel Computing. Keynote. In *ICPP'07*, Sep. 2007.

[7] J. L. Hennessy and D. A. Patterson. *Computer Architecture: A Quantitative Approach*. Fourth edition, 2006.

[8] M. Holland. *On-Line Data Reconstruction in Redundant Disk Arrays*. PhD thesis, Carnegie Mellon University, Apr. 1994.

[9] M. Holland, G. Gibson, and D. P. Siewiorek. Architectures and Algorithms for On-Line Failure Recovery in Redundant Disk Arrays. *Journal of Distributed and Parallel Databases*, 2(3):295–335, Jul. 1994.

[10] IBM Easy Tier. http://www.almaden.ibm.com/storagesystems/projects/easytier.

[11] S. Kang and A. Reddy. User-Centric Data Migration in Networked Storage Systems. In *IPDPS'08*, Apr. 2008.

[12] J. Lee and J. Lui. Automatic Recovery from Disk Failure in Continuous-Media Servers. *IEEE Transactions on Parallel and Distributed Systems*, 13(5):499–515, May. 2002.

[13] [MD PATCH 09/16] md/raid5: preferentially read from replacement device if possible. http://www.spinics.net/lists/raid/msg36361.html.

[14] D. Narayanan, A. Donnelly, and A. Rowstron. Write Off-Loading: Practical Power Management for Enterprise Storage. In *FAST'08*, Feb. 2008.

[15] OLTP Application I/O and Search Engine I/O. http://traces.cs.umass.edu/index.php/Storage/Storage.

[16] D. A. Patterson, G. Gibson, and R. H. Katz. A Case for Redundant Arrays of Inexpensive Disks (RAID). In *SIGMOD'88*, Jun. 1988.

[17] E. Pinheiro and R. Bianchini. Energy Conservation Techniques for Disk Array-Based Servers. In *ICS'04*, Jun. 2004.

[18] E. Pinheiro, W.-D. Weber, and L. A. Barroso. Failure Trends in a Large Disk Drive Population. In *FAST'07*, Feb. 2007.

[19] M. Saxena and Michael M. Swift. FlashVM: Virtual Memory Management on Flash. In *USENIX ATC'10*, Jun. 2010.

[20] B. Schroeder and G. Gibson. Disk Failures in the Real World: What Does an MTTF of 1,000,000 Hours Mean to You? In *FAST'07*, Feb. 2007.

[21] M. Sivathanu, V. Prabhakaran, F. I. Popovici, T. E. Denehy, A. C. Arpaci-Dusseau, and R. H. Arpaci-Dusseau. Improving Storage System Availability with D-GRAID. In *FAST'04*, Mar. 2004.

[22] L. Tian, Q. Cao, H. Jiang, D. Feng, C. Xie, and Q. Xin. SPA: On-Line Availability Upgrades for Parity-based RAIDs through Supplementary Parity Augmentations. *ACM Transactions on Storage*, 6(4), 2011.

[23] L. Tian, D. Feng, H. Jiang, K. Zhou, L. Zeng, J. Chen, Z. Wang, and Z. Song. PRO: A Popularity-based Multi-threaded Reconstruction Optimization for RAID-Structured Storage Systems. In *FAST'07*, Feb. 2007.

[24] S. Wan, Q. Cao, J. Huang, S. Li, X. Li, S. Zhan, L. Yu, C. Xie, and X. He. Victim Disk First: An Asymmetric Cache to Boost the Performance of Disk Arrays under Faulty Conditions. In *USENIX ATC'11*, Jun. 2011.

[25] C. Weddle, M. Oldham, J. Qian, A. Wang, P. Reiher, and G. Kuenning. PARAID: A Gear-Shifting Power-Aware RAID. In *FAST'07*, Feb. 2007.

[26] B. Welch, M. Unangst, Z. Abbasi, G. Gibson, B. Mueller, J. Small, J. Zelenka, and B. Zhou. Scalable Performance of the Panasas Parallel File System. In *FAST'08*, Feb. 2008.

[27] S. Wu, D. Feng, H. Jiang, B. Mao, L. Zeng, and J. Chen. JOR: A Journal-guided Reconstruction Optimization for RAID-Structured Storage Systems. In *ICPADS'09*, Dec. 2009.

[28] S. Wu, H. Jiang, D. Feng, L. Tian, and B. Mao. WorkOut: I/O Workload Outsourcing for Boosting RAID Reconstruction Performance. In *FAST'09*, Feb. 2009.

[29] S. Wu, B. Mao, D. Feng, and J. Chen. Availability-Aware Cache Management with Improved RAID Reconstruction Performance. In *CSE'10*, Dec. 2010.

[30] T. Xie and H. Wang. MICRO: A Multilevel Caching-Based Reconstruction Optimization for Mobile Storage Systems. *IEEE Transactions on Computers*, 57(10):1386–1398, 2008.

[31] Q. Xin, E. L. Miller, and T. J. E. Schwarz. Evaluation of Distributed Recovery in Large-Scale Storage Systems. In *HPDC'04*, Jun. 2004.